Come funziona Linux:

Installazione di software e gestione del software installato

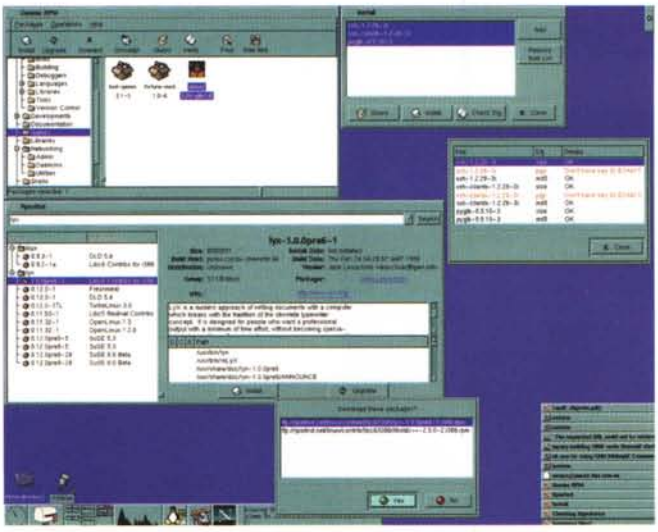
Linux permette di installare nuovi programmi in modo semplice e con il pieno controllo delle dipendenze fra i diversi pacchetti installati: ogni cosa viene inserita nel sistema in modo logico e alla fine tutti i pezzi si trovano dove ci si aspetta di trovarli. Una volta che abbiamo un sistema funzionante dobbiamo ricordarci di fare il backup... vedremo come farlo il mese prossimo.

Ottava parte

di Giuseppe Zanetti

Mai più dunque decine di librerie tutte nella stessa directory, di cui non è possibile ricostruire la storia in quanto installate da programmi che magari sono stati tolti dal sistema da mesi.

Il programma di gestione dei pacchetti rpm, utilizzato ormai da quasi tutte le distribuzioni di Linux, permette infatti di installare e disinstallare software in modo assai semplice, avvertendo e chiedendo conferma all'utente prima di effettuare operazioni critiche e permettendo in ogni momento di verificare lo stato del software installato. Anche la disinstallazione avviene in modo assai pulito, portandosi via tutto senza lasciare porcheria nel filesystem.



Gli utenti di Gnome possono gestire in modo semplice i pacchetti RPM usando il programma GnoRPM.

Ovviamente l'installazione di software di sistema in Linux non può essere fatta da chiunque, ma solamente dal superutente "root" (ciò non vieta tuttavia di installarsi una copia personale di certi programmi in una propria directory).

Questa potrebbe apparire a prima vista una complicazione, ma ci si accorgerà che in realtà è un modo di fare le cose che alla lunga porta i propri frutti, permettendo di avere un sistema sempre stabile e sicuro, senza il rischio che operazioni errate compiute da un normale utente possano compromettere tutto il sistema.

Il sistema RPM

L'avvento di distribuzioni di Linux contenenti decine, se non centinaia, di programmi, ha fatto sentire, alcuni anni or sono, la necessità di disporre di un sistema per installare e rimuovere facilmente dei pacchetti software, verificando eventuali dipendenze e correlazioni fra i pacchetti installati. Inizialmente ogni distribuzione faceva le cose a modo proprio: Slackware utilizzava pacchetti in formato tar compressi con gzip, Debian e Red Hat un proprio formato proprietario. E così via per le altre distribuzioni minori. Alla fine è prevalso come standard "de facto" il formato RPM (Red Hat Package Manager, estesione dei file: .rpm) di Red Hat, che ora è utilizzabile, in modo nativo o mediante appositi strumenti di conversione, da quasi tutte le distribuzioni. Ciò non significa automaticamente che un pacchetto software pensato per Red Hat possa tranquillamente essere installato in un sistema Linux di SuSe, o viceversa, in quanto potrebbe dipendere da librerie o altri programmi non installati da questa distribuzione. Oppure potrebbe anche essere che il pacchetto contenga dei percorsi errati per la distribuzione su cui lo si tenta di installare, anche se questa eventualità in teoria non dovrebbe sussistere, in quanto quasi tutte le distribuzioni si

sono adeguate ad uno stesso standard per disposizione dei file nel filesystem.

Se state utilizzando un sistema Red Hat, i pacchetti in formato RPM contenenti il software distribuito di serie si trovano nel CD di installazione all'interno della directory RedHat/RPMS.

Per gestire il database del software installato si utilizza il comando rpm. Esso ha le seguenti funzioni principali:

- ✔ installazione ed upgrade di un pacchetto .rpm
- ✔ rimozione di un pacchetto installato
- ✔ gestione delle dipendenza fra pacchetti
- ✔ gestione del database dei pacchetti (/var/lib/rpm/packa ges.rpm)
- ✔ creazione di pacchetti

Ulteriori informazioni su RPM si possono trovare nel manuale in linea oppure nel sito http://www.rpm.org/.

Installazione di un pacchetto usando RPM

Installare un nuovo software disponibile in formato RPM è veramente semplice. Ovviamente per compiere l'operazione occorrono i permessi di root.

```
# rpm --install /RedHat/RPMS/apache-1.3.9-4.i386.rpm
```

Il pacchetto da installare può risiedere all'interno del filesystem della macchina oppure su un server HTTP o FTP remoto. In questo caso il percorso da specificare deve iniziare con http:// oppure ftp://, seguendo la convenzione tipica degli URL. In questi casi rpm si occupa automaticamente di collegarsi al sito richiesto e di scaricare il pacchetto. Nel caso si fosse nascosti dietro un proxy, è possibile istruire opportunamente rpm mediante le opzioni --ftpproxy e --httpproxy (tali configurazioni possono essere salvate una volta per tutte nel file /etc/rpmrc).

```
# rpm --ftpproxy proxy.profuso.com --install
  ftp://ftp.redhat.com//pub/contrib/libc6/i586/w3m-1-1.i586.rpm
```

Nel caso il pacchetto risultasse già installato, si otterrà il messaggio:

```
package apache-1.3.9-4 is already installed
```

Un'altra causa di non installazione di un pacchetto è la dipendenza dello stesso da altri pacchetti non ancora installati nel sistema:

```
# rpm --install /RedHat/RPMS/ucd-snmp-utils-4.0.1-4.i386.rpm
error: failed dependencies:
  ucd-snmp is needed by ucd-snmp-utils-4.0.1-4
  libsnmp.so.0 is needed by ucd-snmp-utils-4.0.1-4
```

Per risolvere le dipendenze si possono installare i vari pacchetti mancanti (si usi l'opzione --query --whatprovides per scoprire da quale pacchetto deriva una determinata dipendenza) e riprovare ad installare il pacchetto che presenta problemi:

```
# rpm --install /RedHat/RPMS/ucd-snmp-4.0.1-4.i386.rpm
# rpm --install /RedHat/RPMS/ucd-snmp-utils-4.0.1-4.i386.rpm
```

In alternativa, qualora le dipendenze non siano di troppo conto, si può forzare l'installazione del pacchetto mediante l'opzione --nodeps:

```
# rpm --install --nodeps /RedHat/RPMS/ucd-snmp-utils-4.0.1-4.i386.rpm
```

In quest'ultimo caso è però possibile che il programma installato non funzioni correttamente quando si tenta di avviarlo.

Elenco del software installato

Per ottenere la lista dei pacchetti installati si utilizza l'opzione --query -a di rpm:

```
# rpm --query -a
setup-2.0.5-1
filesystem-1.3.5-1
basesystem-6.0-4
...
wine-20000821-1
ucd-snmp-utils-4.0.1-4
```

La stessa opzione, usata con lo switch -l, permette di conoscere il contenuto di un pacchetto installato (si noti che in questo caso si indica il nome del pacchetto e non il percorso del file .rpm):

```
# rpm --query -l xcdroast-0.96ex-1
/usr/bin/xcdroast
/usr/doc/xcdroast-0.96ex
/usr/doc/xcdroast-0.96ex/COPYING
/usr/doc/xcdroast-0.96ex/Changes
/usr/doc/xcdroast-0.96ex/README.ATAPI
...
```

Nel caso si desideri invece conoscere il contenuto di un pacchetto non ancora installato bisogna specificare il percorso e aggiungere l'opzione -p:

```
# rpm --query -l -p /RedHat/RPMS/xsysinfo-1.7-1.i386.rpm
```

Analisi delle caratteristiche di un pacchetto

La funzione --query ha altre opzioni interessanti: permette ad esempio di conoscere le caratteristiche di un pacchetto, come nome dell'autore, licenza, descrizione, ...

```
# rpm --query -i apache-1.3.9-4
Name    : apache  Relocations: (not relocateable)
Version : 1.3.9   Vendor: Red Hat Software
```

```
Release: 4        Build Date: Tue 1 Sep 1999
Install date:(not installed)  Build Host:
porky.devel.redhat.com
Group : System Daemons Source RPM: apache-1.3.9-
4.src.rpm
Size  : 2439994  License: Freely distributable
Packager: Red Hat Software
Summary : The most widely used Web server on the
Internet.
Description :
Apache is a powerful, full-featured, efficient and
freely-available Web server. Apache is also the
most popular
Web server on the Internet.
Install the apache package if you need a Web
server.
```

Se il pacchetto non è ancora installato bisogna, al solito, usare l'opzione -p seguita dal nome del file.

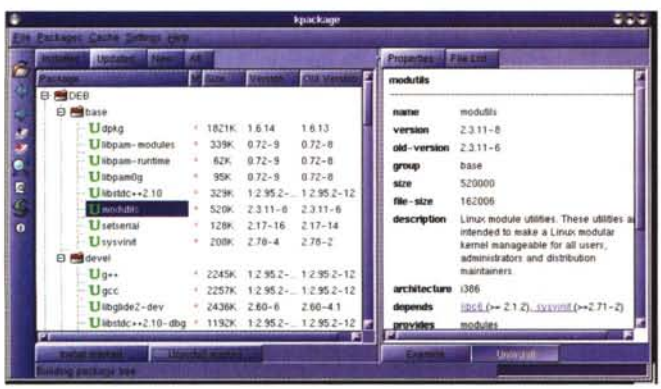
```
# rpm --query -i -p /RedHat/RPMS/apache-1.3.9-
4.i386.rpm
```

Ulteriori opzioni di rpm permettono di conoscere da quali componenti dipende il pacchetto:

```
# rpm --query --requires -p apache-1.3.9-
4.i386.rpm
/etc/mime.types
/sbin/chkconfig
/bin/mktemp
/bin/rm
...
/bin/sh
/usr/bin/perl
libc.so.6(GLIBC_2.0)
```

e quali esso fornisce

```
# rpm --query --provides -p apache-1.3.9-
4.i386.rpm
webserver
libproxy.so
mod_access.so
...
mod_vhost_alias.so
```



Per l'ambiente KDE è invece disponibile l'interfaccia grafica Kpackage.

Upgrade di un pacchetto

E` possibile eseguire l'aggioramento di un pacchetto mediante la funzione -U:

```
# rpm -U /cdrom/RedHat/RPMS/apache-1.3.9-
4.i386.rpm
```

Questo comando funziona in modo analogo a --install, con la differenza che rimuove l'eventuale vecchia versione del pacchetto qualora esso risultasse già installato. Ciò significa la perdita dei file di configurazione. L'analoga opzione --freshen invece rispetta i pacchetti già presenti:

```
# rpm --freshen /cdrom/RedHat/RPMS/apache-1.3.9-
4.i386.rpm
```

Eventuali file di configurazione derivanti da versioni precedenti vengono salvati col suffisso .rpmsave, in modo da non perdere le modifiche eventualmente effettuate.

Anche in questo caso è possibile prelevare direttamente il pacchetto da installare mediante FTP o HTTP.

Controllo di integrità

Nel caso di "incidenti" alla propria installazione di Linux, è possibile controllare l'integrità del software installato mediante l'opzione --verify (abbreviabile in -V):

```
# rm /usr/X11R6/man/man1/xmessage.1x
# rpm --verify X11R6-contrib-3.3.2-6
missing     /usr/X11R6/man/man1/xmessage.1x
```

La verifica di integrità non si limita solamente alla lista di file contenuti nel pacchetti, ai loro permessi ed alla loro lunghezza, ma utilizza tecniche di firma crittografica (MD5) per essere sicuri al 100% che il pacchetto non sia stato modificato.

Alcuni pacchetti risultano firmati mediante PGP (lo stesso software usato per la firma dei messaggi di e-mail), il che permette di certificarne la provenienza e l'integrità. Mediante rpm è possibile verificare tale firma:

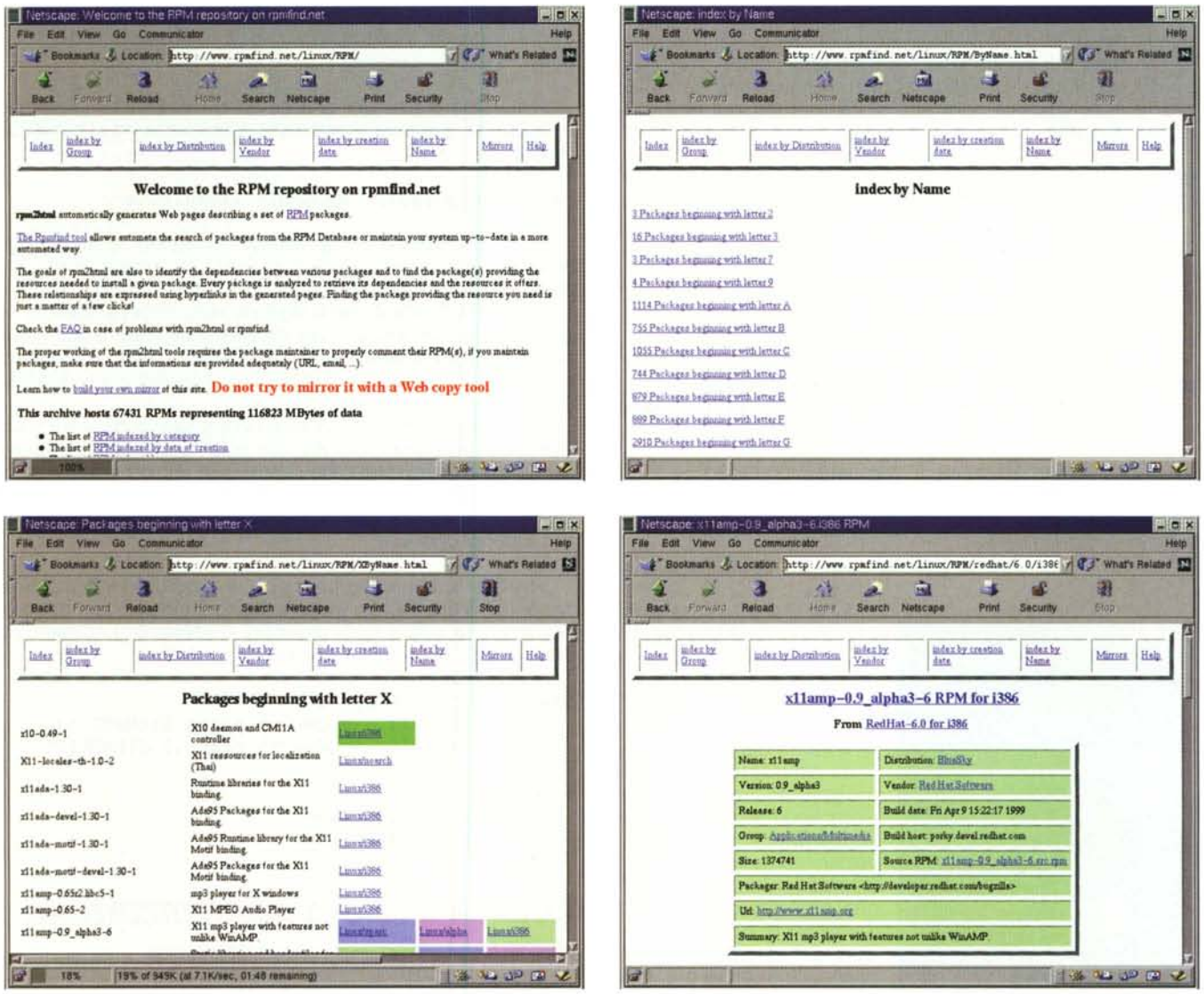
```
# rpm --checksig apache-1.3.9-4.i386.rpm
apache-1.3.9-4.i386.rpm: md5 GPG NOT OK
```

Rimozione di un pacchetto

Per eliminare un pacchetto si utilizza la funzione --erase di rpm. Anche in questo caso si è soggetti al controllo delle dipendenze (forse a maggior ragione, in quanto serve ad evitare di avere un sistema con componenti mancanti):

```
# rpm --erase ucd-snmp-4.0.1-4
error: removing these packages would break de-
pendencies:
    ucd-snmp is needed by ucd-snmp-utils-4.0.1-4
    libsnmp.so.0 is needed by ucd-snmp-utils-
4.0.1-4
```

Per eliminare il pacchetto bisogna prima eliminare quelli



I vari passi della ricerca di un pacchetto con rpmfind

che gli causano problemi, operando in senso contrario alle dipendenze.

```
# rpm --erase ucd-snmp-utils-4.0.1-4
# rpm --erase ucd-snmp-4.0.1-4
```

Compilazione di un programma a partire dal codice sorgente

Dopo aver descritto il funzionamento e la potenza di RPM, vediamo anche come è possibile installare programmi direttamente a partire dal codice sorgente. Non tutti i programmi infatti sono disponibili come file RPM e comunque può esserci la necessità di ricompilare un software, ad esempio per abilitare particolari funzioni non disponibili di serie.

L'operazione di compilazione di un programma è comunque facilitata, rispetto al passato, dalla presenza in Linux di una serie di ottimi tool, frutto del progetto GNU, che permettono di automatizzarla. Tali strumenti sono tuttavia d'aiuto solamente se il programmatore si è preso la briga di utilizzarli nel proprio lavoro.

Essendo nella maggior parte dei casi i programmi pensati per essere utilizzati su vari dialetti di UNIX (o release o installazioni dello stesso sistema contenenti librerie diverse o più o meno aggiornate), tradizionalmente la compilazione di un programma richiedeva la lettura accurata dei file README inclusi con i sorgenti e la "customizzazione" del Makefile in base alle caratteristiche presenti nel proprio particolare sistema.

Makefile è il file di configurazione utilizzato dal programma make per descrivere i file necessari per compilare un particolare programma. Generalmente esso contiene una lista di di-

pendendze che descrivono in quale ordine compilare i file. Ciò permette di evitare di ricompilare ogni volta anche i sorgenti che non sono stati modificati. Il seguente semplice esempio di Makefile permette di capire meglio il funzionamento di make:

```
CC=gcc
CFLAGS=-static -DDEBUG

INSTDIR=/usr/local
BINDIR=/usr/local/bin
MANDIR=/usr/local/man

all:    pippo

pippo:  pippo.o util.o now.o arpa_date.o
        $(CC) -o pippo pippo.o util.o now.o
reg.o -lcrypt

pippo.o: pippo.c pippo.h upload.o libmulti.o
        $(CC) -o pippo.o  upload.o libmulti.o

%.o :   %.c config.h
        $(CC) $(CLAFGS) -c  $< -o  $@ $(INCLUDE)
```

Nonostante la quantità di dati gestiti ed il numero di visitatori, rpmfind.net è gestito da un normalissimo PC con Red Hat Linux

Si noti che la spaziatura all'interno del Makefile avviene utilizzando il carattere di tabulazione TAB.

Le macro CC e CFLAGS definiscono rispettivamente il nome del compilatore C da utilizzare (molti dei programmi per UNIX e Linux sono scritti in questo linguaggio) e le eventuali opzioni con cui richiamarlo. Queste macro, se non definite dall'utente, prendono dei valori di default dipendenti dal sistema che si sta usando (nel caso di Linux spesso CC contiene il valore "gcc", che indica di utilizzare il compilatore GNU C). INSTDIR, BINDIR e MANDIR, che ci serviranno dopo, sono delle macro definite dal programmatore che indicano dove installare i vari componenti del programma.

La prima riga successiva alle definizioni indica l'operazione di default da compiere quando make viene richiamato senza parametri (etichetta all), in questo caso la generazione del file pippo. In base alle regole elencate di seguito, esso dipende dai file pippo.o util.o now.o arpa_date.o e il comando per ottenerlo è

```
$(CC) -o pippo pippo.o util.o now.o reg.o -
lcrypt
```

il quale, una volta sostituite le macro, diventa:

```
gcc -static -DDEBUG -o pippo pippo.o util.o
now.o reg.o -lcrypt
```

A questo punto make si accorge che manca il file pippo.o e lo tenta di generare in base alle dipendenze ed al comando relativi:

```
$(CC) -o pippo.o  upload.o libmulti.o
```

Lo stesso accade per gli altri file mancanti. L'ultima riga definisce una regola di default da usare per creare un file .o a partire dal suo sorgente (file .c).

Il funzionamento di make dipende dalla data dei vari file. In questo modo se nessuno dei file da cui dipende pippo è stato modificato più recentemente di pippo stesso, il programma viene considerato aggiornato ("up to date") e non viene ricompilato.

In questo modo non occorre ricompilare tutto ogni volta che si apporta una modifica, ma solo quelli che dipendono dal file modificato. Nel caso occorresse aggiornare la data di un file si potrà usare il comando touch:

```
# touch pippo.c
```

Volendo fare le cose per bene, è possibile definire delle sezioni di Makefile che automatizzino l'installazione del programma e la pulizia della directory contenente i sorgenti da eventuali file generati dal compilatore:

```
install: pippo pippo.1
         install --mode=755 pippo $(INSTDIR)
         install --mode=755 pippo.1 $(MANDIR)

clean:
        rm -f *.o *.bak
        rm -f pippo
```

Automatizzare la customizzazione dei Makefile con configure

E` frequente trovare, all'interno dei sorgenti C di programmi compilabili sotto diverse versioni di UNIX, parecchie righe che indicano al compilatore di includere o meno certe parti del codice a seconda che si stia utilizzando un certo sistema operativo o si abbiano installate certe librerie piuttosto che altre:

```
#ifdef LINUX
       parte del codice specifica per Linux
#endif

#ifdef SOLARIS
       parte del codice specifica per Solaris
#endif

#ifdef HAVE_STRCMP
       codice compilato solo se si dispone
       della funzione strcmp()
#else
       codice compilato solo se non si dispone
```

```
        della funzione strcmp()
#endif
```

Generalmente le macro LINUX, SOLARIS, HAVE_STRCMP vengono definite automaticamente dal compilatore oppure "a mano" all'interno del Makefile, come opzioni di tipo CFLAGS (es: -DHAVE_STRCMP), tuttavia nel caso il programma sia complesso e dipenda da parecchie librerie, tale operazione diventa abbastanza complessa.

Per questo, nell'ambito del progetto GNU, è stato scritto il tool configure, il quale si occupa di fare un inventario delle caratteristiche del sistema e delle librerie installate e di creare automaticamente il Makefile. Nei programmi, sempre più diffusi, che fanno uso di questo strumento, la compilazione diventa un'operazione molto semplice, in quanto è sufficiente eseguire lo script configure presente nella direcotry corrente (perciò lo si esegue in modo relativo ad essa: ./configure) e nell'avviare make. Nel caso di programmi complessi con diverse opzioni di compilazione, è possibile scegliere cosa includere nel risultato finale specificandolo nella linea di comando di configure.

```
# cd php-3.0.16
# ./configure --with-mysql
loading cache ./config.cache
...
checking for bison... (cached) bison -y
checking bison version... 1.28 (ok)
checking for gcc... (cached) gcc
checking whether the C compiler (gcc  ) works...
yes
checking whether the C compiler (gcc  ) is a
cross-compiler... no
checking whether we are using GNU C... (cached)
yes
checking whether gcc accepts -O2... (cached) yes
...
checking for fcntl.h... (cached) yes
checking for unistd.h... (cached) yes
checking for crypt.h... (cached) yes
checking for sys/file.h... (cached) yes
...
checking for MySQL support... yes
...
creating ./config.status
creating Makefile
```

Nell'esempio appena visto, ho compilato il linguaggio PHP3 specificando nella linea di comando di configure che desideravo il supporto per il database relazionale MySQL. Per prima cosa lo script ha verificato che nella macchina fosse installato un compilatore C funzionante, trovando il GCC (GNU C Compiler) e ha cercato quali eventuali parametri fossero utilizzabili con esso, fra cui -O2, il quale è un selettore che indica il livello di ottimizzazione del codice oggetto generato.

In seguito sono stati controllati, uno per uno, i diversi "include file" e librerie necessari alla compilazione del programma per vedere che fossero tutti presenti nel sistema. I programmi ben scritti per quanto riguarda la portabilità fra piattaforme diverse prevedono il supporto di più modi per compiere la stessa opperazione nel caso il sistema target non

rpmfind.net

Se non volete cimentarvi nella ricompilazione di un programma e nella sua homepage non riuscite a trovare il pacchetto .rpm corrispondente, non disperate, ma provate a collegarvi al sito http://www.rpmfind.net/ o ad uno dei suoi mirror.
Si tratta di un vastissimo database di titoli di programmi disponibili in formato RPM. Nel momento in cui scrivo (Settembre 2000), l'archivio comprende oltre 67000 titoli. Non si tratta, purtroppo, di un database, bensì di una serie di pagine statiche HTML generate mediante il programma rpm2html. La lista è navigabile alfabeticamente, per piattaforma (x86, alpha, ...) oppure per origine del file (dalle maggiori distribuzioni o dai loro "contrib"). Infine i programmi sono distinti per il fatto che facciano uso della libc5 o della più recente glibc6. Per ogni programma vengono listati tutti i dati ottenibili mediante le diverse opzioni della funzione –query di rpm, in modo che si sappia prima di installare il programma cosa verrà caricato sul disco e di quali file o pacchetti aggiuntivi ci sarà bisogno. Un semplice click col tasto destro del mouse sarà poi sufficiente per scaricare il pacchetto .rpm desiderato nel proprio PC.
Per chi volesse avere la potenza di rpmfind sempre a portata di mano senza doversi ogni volta collegare al sito, è possibile installare il programma rpmfind. Si tratta di un client a linea di comando che permette di ricercare nella lista direttamente dalla propria sessione di lavoro usando una parola chiave o una espressione regolare. Ad esempio per cercare tutti i pacchetti RPM la cui descrizione contenga la parola Borland, si utilizza il comando

```
# rpmfind –apropos borland
1: ftp://rpmfind.net/linux/contrib/i386/rhide-1.3-1.i386.rpm
  rhide : Rhide is a very nice IDE exactly like Borland's
```

Volendo, è possibile fare in modo che rpmfind cerchi, prelevi ed installi in un unico passaggio il programma desiderato:

```
# rpmfind xbill
Arch : i586, Os : Linux
Default distribution : Red Hat Software(Hurricane)
owning 249 of 338 installed packages
Get http://rpmfind.net//linux/RDF/resources/xbill.rdf
Get http://rpmfind.net//linux/RDF/redhat/5.0/i386/xbill-2.0-2.i386.rdf
Installing xbill will requires 183 KBytes

### To Transfer:
ftp://rpmfind.net/linux/redhat/i386/RedHat/RPMS/xbill-2.0-
2.i386.rpm
Do you want to download these files to /tmp [Y/n/a] ? : y
saving to /tmp/xbill-2.0-2.i386.rpm

# rpm –install /tmp/xbill-2.0-2.i386.rpm
```

Nel caso il programma desiderato dipendesse da altri file, rpmfind si occuperà di prelevarli tutti.
Un altro possibile uso interesante è il prelievo della versione più aggiornata di un pacchetto:

```
$ rpmfind -$ rpmfind -q –latest knews
```

o l'upgrade di un pacchetto:

```
$ rpmfind -q –upgrade balsa
```

Rpmfind è disponibile presso il sito http://rpmfind.net/linux/rpm2html/rpmfind.html ed è fornito di serie con le maggiori distribuzioni di Linux.

preveda certe librerie o funzioni. Nel caso specifico viene controllata anche la presenza del database MySQL.

Il risultato di configure sono uno o più Makefile già configurati in base alle caratteristiche presenti nel sistema ed alle richieste dell'utente.

A questo punto per completare l'opera è sufficiente dare un bel "make" e, dopo una buona tazza di caffè (PHP è un programma abbastanza complesso), un "make install":

```
# make
gcc   -g   -O2   -I/mysql-3.22.32/include   -c
parser.tab.c -o parser.tab.o
...
# make install
...
```

Esempio: compilazione di PHP3

Ricapitolando, per compilare un programma le operazioni da compiere sono: innanzitutto la scompattazione dell'archivio contenente i sorgenti in una directory temporanea.

Nel caso l'archivio sia in formato tar.gz si possono utilizzare in sequenza i comandi "gzip -d" (scompatta il file .tar.gz in un file archivio .tar) e "tar" per scompattare l'archivio stesso:

```
# gzip -d php-3.0.16.tar.gz
# tar xvf php-3.0.16.tar
php-3.0.16/
php-3.0.16/acconfig.h
php-3.0.16/aclocal.m4
php-3.0.16/alloca.c
...
# cd php-3.0.16
```

A questo punto, dopo una doverosa lettura dei vari file informativi (README, INSTALL, ...) si può usare configure per creare i Makefile necessari e make per compilare ed installare il software, come abbiamo già visto in precedenza.

Archiviare file in Linux: Tar, compress, gzip e bzip2

Parlando di archivi, non possiamo non vedere come funziona l'archiviazione di file in Linux, in particolare usando il comando tar.

Oltre ai sorgenti dei programmi, anche alcuni pacchetti precompilati vengono forniti come archivi .tar da scompattare nel proprio sistema. Nella prossima puntata metteremo inoltre a frutto le nozioni ora imparate anche per gestire il backup della nostra macchina.

Il programma standard per l'archiviazione di file in UNIX è "tar", anche se sono disponibili versioni di tutti gli archiviatori più famosii (zip, lha, ...). Tar ha il pregio, rispetto a questi ultimi, di mantenere integre le informazioni sulla proprietà e i permessi dei file anche passando l'archivio fra versioni diverse di UNIX.

La versione fornita con Linux deriva, al solito, dal progetto GNU ed è compatibile con gli archivi creati dagli altri sistemi UNIX, pur offrendo diverse funzioni utili aggiuntive, come il supporto per creare archivi multivolume (ad esempio su più dischetti o nastri). Al contrario di programmi come zip, tar esegue solo l'archiviazione dei file, senza comprimere il risultato. Vedremo che tale funzione può essere eseguita in seguito usando sull'archivio creato altri programmi, come compress, bzip2 o gzip (quest'ultimo è lo standard sotto Linux).

Per creare un archivio in Linux si utilizza l'opzione "-c" di tar, nel seguente modo:

```
# tar -c elenco di file o direc-
tory
```

In questo caso l'archivio viene creato nel nastro di sistema, corrispondente al file speciale /dev/tape. Nel caso non si disponga di un nastro di backup, è possibile specificare, mediante l'opzione "-f", il nome di un file in cui inserire l'archivio.

Nell'esempio che segue si è anche utilizzato "-v" per ottenere una lista sullo

Meeting del Pluto a Terni

Mi comunicano gli amici del PLUTO, che il tradizionale incontro degli utenti italiani di Linux, dopo tre anni tornerà in Umbria, a Terni, dal 9 all'11 dicembre si terrà il Pluto Meeting 2000, una manifestazione dedicata alla diffusione del Software Libero.

L'evento comprenderà una serie di interventi e lezioni sull'uso del Software Libero nelle aziende e per la produttività individuale, una serie di incontri su temi proposti dai partecipanti ed una zona di esposizione dove le aziende che investono in questa branca dell'informatica potranno incontrare gli utenti.

Il Centro MultiMediale, dove si svolgerà l'evento, è composto da due strutture, il Videocentro e la Bibliomediateca, distanti 10 minuti a piedi. Gli indirizzi sono, rispettivamente:

- ✔ Piazzale Bosco (a 5 minuti a piedi, forse meno, dalla stazione ferroviaria e dalla stazione degli autobus)
- ✔ Piazza della Repubblica 1 (a 10/15 minuti a piedi dalla stazione ferroviaria)

All'evento prenderanno parte personalità del calibro di Andrea Arcangeli, sviluppatore del Kernel per la SuSE, Wichert Akkerman leader della distribuzione Debian e Carlo Daffara, membro del Gruppo di lavoro sul Software Libero dell' Information Society Directorate General della Comunità Europea.

Ulteriori informazioni possono essere reperite nel sito http://meeting.pluto.linux.it/.

schermo dei file mano a mano che essi vengono inseriti nell'archivio.

```
# tar -cvf nomefile.tar elenco di file o direc-
tory
```

Nel caso vengano specificate delle directory, vengono inseriti in archivio tutti i file in esse contenuti, in modo ricorsivo.

Tar è l'unico programma in cui, per ragioni storiche, è opzionale l'utilizzo del simbolo - per indicare le opzioni. La linea precedente può perciò anche essere tranquillamente riscritta come:

```
# tar cvf nomefile.tar elenco di file o direc-
tory
```

Una volta creato l'archivio, si può vederne il contenuto usando l'opzione -t.

```
# tar -tf nomefile.tar
etc/
etc/exports
etc/group
...
```

In questo caso se si aggiunge anche -v, il risultato viene fornito in modo più dettagliato, simile a "ls -l":

```
# tar -tvf nomefile.tar
drwxr-xr-x root/root              0 2000-09-04
18:58:15 etc/
-rw-r--r-- root/root            389 2000-05-01
17:59:53 etc/exports
-rw-r--r-- root/root            591 2000-07-31
10:44:06 etc/group
...
```

Per recuperare l'archivio ci si posiziona nella directory da cui lo si vuole recuperare e si utilizza l'opzione -x:

```
# tar -xf nomefile.tar
```

anche in questo caso può essere usato il -v per vedere cosa sta facendo tar.

I percorsi dei file vengono salvati in un archivio in formato tar in modo relativo alla directory corrente (es: etc/passwd, senza la / iniziale), in modo da minimizzare il pericolo di rovinare file di sistema estraendoci sopra un archivio e per permettere di recuperare un archivio in qualunque posizione del filesystem, ad esempio sotto /tmp:

```
# cd /tmp
# mkdir miobackup
# cd miobackup
# tar xvf /cdrom/backup.20000812.tar
```

Archivi compressi

Generalmente per ottenere un archivio compresso si parte dall'archivio con estensione .tar e lo si comprime utilizzando uno dei compressori disponibili

Programma	estensione	per comprimere	per decomprimere
compress	.Z	compress nomefile	compress -d nomefile
gzip	.gz oppure .z	gzip nomefile	gzip -d nomefile
bzip2	.bz2	bzip2 nomefile	bzip2 -d nomefile

In ordine di efficienza il migliore è bzip2, seguito da gzip. Quello standard nei sistemi Linux è gzip (esso non è disponibile di serie in tutte le versioni di UNIX, perciò per scambiare file con esse conviene usare compress).

Gzip è in grado di decomprimere senza problemi file creati usando compress, mentre, ovviamente, non vale il viceversa. Bzip2 è un programma relativamente nuovo e perciò ancora poco utilizzato, nonostante le sue capacità.

Tornando al nostro file .tar, le operazioni da compiere sono le seguenti (l'opzione -9 di tar crea un archivio più compatto, a costo di una maggior lentezza nella codifica):

```
# tar -cf nomefile.tar lista di file
# gzip -9 nomefile.tar
```

Per scomprimere l'archivio si usa invece:

```
# gzip -d nomefile.tar.gz
# tar -xvf nomefile.tar
```

La versione di tar fornita con Linux (GNU tar) permette di creare o leggere in modo automatico archivi compressi con gzip, senza dover compiere separatamente le due operazioni, specificando l'ozione -z.

```
# tar -xzf archivio.tar.gz
```

Opzioni analoghe permettono di lavorare per gli archivi in formato compress (-Z) e bzip2 (-I):

```
# tar -xIf archivio.tar.bz2
```

Le funzioni possibili in tar non si limitano certamente solo a quelle appena viste. Vedremo nelle prossime puntate che esso è uno strumento completo e potente, che permette di creare in modo semplice archivi e copie di backup del proprio sistema, anche in modo automatico o incrementale.

Conclusioni

In questa puntata abbiamo incontrato molti aspetti interessanti di Linux. Nonostante installare software usando una interfaccia grafica possa sembrare più semplice, in realtà abbiamo visto che per fare le cose per bene si tratta solamente di digitare un'unica semplice riga di comando.

Alcune distribuzioni di Linux forniscono all'utente dei metodi grafici per installare il software, che in realtà sono dei frontend sopra rpm.

Esiste anche un modulo di Linuxconf (vedere MC di Settembre) che consente di installare e gestire pacchetti RPM. Nel caso si fosse interessati, si può trovare la lista di tali software su http://www.rpm.org/software.html.

MC